

Algoritmos de Aceleração

Marcelo Walter - Unisinos

Como acelerar?

- ◆ Estruturas de dados espaciais eficientes
- ◆ *Culling techniques*
 - Lidar apenas com o necessário (eg, Portais)
- ◆ Geometria adaptativa - Níveis de detalhe (próxima aula)

Estruturas de Dados Espaciais

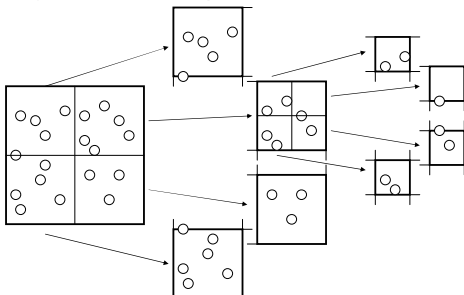
- ◆ Organizam a geometria através da sua localização no espaço
 - Por exemplo, armazenar pontos de acordo com sua localização, ou polígonos,...
- ◆ “Multitude” de usos em jogos
 - Visibilidade - O que consigo ver daqui?
 - Intersecções de raios - Qual objeto foi atingido por aquele tiro da Lara Croft?
 - Detecção de Colisões - O jogador atingiu um muro?
 - Proximidade - Quem é afetado por esta explosão?

Decomposição do Espaço

- ◆ Como responder eficientemente as perguntas do slide anterior?
 - Eliminando a maior parte dos objetos rapidamente
- ◆ Estruturas que dividem o espaço em regiões, ou células de algum tipo, hierarquicamente
 - ◆ Árvores que armazenam quem pertence a que região
 - ◆ Uma região pai contém completamente todas as regiões de seus filhos
 - ◆ Se uma query falha para a região, ela falha também para todos os filhos da região
 - ◆ Se a query não falha, indagar os filhos recursivamente
 - ◆ Ao chegar a um nodo folha, fazemos a query somente para os objetos armazenados nesta região

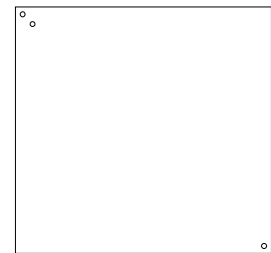
Octrees

- ◆ *Octrees* : planos alinhados com os eixos, regularmente espaçados em cubos



Problemas com Octrees

- ◆ Desbalanceamento se os objetos não estão uniformemente distribuídos (muitos nós sem objetos)
- ◆ Decorrente da subdivisão sempre ao meio...

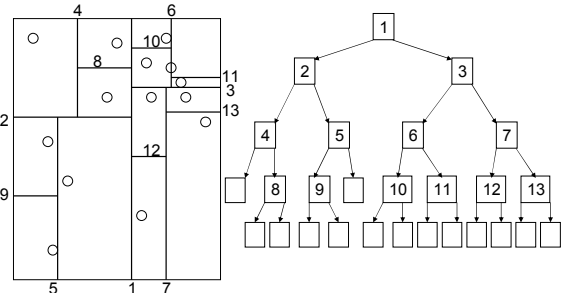


A bad octree case

Kd-Trees

- ◆ *Kd-trees*: planos alinhados com os eixos, em direções alternadas (x,y,z), cortam o espaço em regiões retangulares

Exemplo



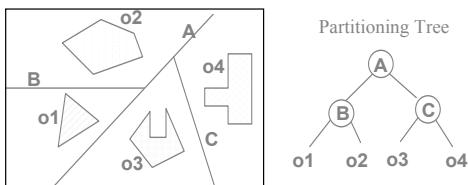
Aplicações

- ◆ Kd-trees funcionam melhor quando os cortes em planos alinhados organizam a informação significativamente
 - eg, ambientes arquitetônicos
- ◆ Tudo que pode ser feito com octrees pode também ser feito com kd-trees
 - eg, View frustum culling, fast ray intersections,...
- ◆ Aplicações específicas:
 - Soda Hall Walkthrough project (Teller and Sequin)
 - ◆ Splitting planes came from large walls and floors
 - Real-time Pedestrian Rendering (University College London)

Binary Space Partitioning Trees

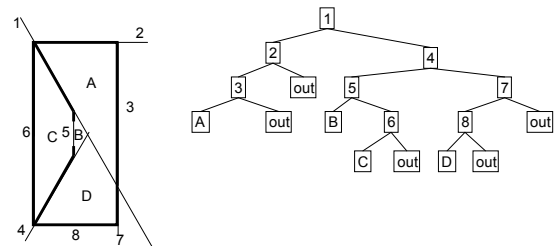
- ◆ *BSP Trees*: Planos alinhados arbitrariamente. Dividem o espaço em regiões convexas
 - Generalização de kd-trees. Uma kd-tree também é conhecida por BSP com os eixos alinhados (axis aligned BSP tree)
- ◆ Divide o espaço em regiões convexas
- ◆ Padrão para subdivisão espacial em jogos
 - Geral o suficiente para lidar com ambientes standard em jogos

Exemplo de BSP



- Os planos de divisão terminam quando interseccionam os planos do nó pai
- Nós internos tem o nome dos planos; as folhas têm o nome de regiões (ou objetos)

Exemplo de BSP



Construindo uma BSP

- ◆ Pré-processamento, pode levar tempo...
- ◆ A maneira mais fácil utiliza os planos definidos pelos polígonos
 - Eq do plano $Ax+By+Cz+D=0$ pode ser derivada a partir das coordenadas do polígono
 - Eq. do plano permite avaliar, facilmente, se um vértice que define o polígono está a frente, atrás, ou no plano

Construindo uma BSP

- ◆ Inicia-se pela raiz com o primeiro polígono da base de dados
- ◆ Os demais polígonos são classificados em relação a este polígono (frente, atrás, cortados)
- ◆ Polígonos que são interseccionados pelo plano são divididos em dois
- ◆ Para cada subespaço criado, um novo polígono é escolhido e assim recursivamente
- ◆ Quando parar? Critério de profundidade ou número de objetos numa região atinge um mínimo

Escolhendo os planos

- ◆ Objetivos:
 - Árvores com poucas regiões
 - Objetos não serem "cortados ao meio" entre regiões
- ◆ Algumas heurísticas:
 - Escolher planos que são planos dos polígonos
 - Escolher polígonos maiores primeiro
 - Escolher planos que não dividem muitos polígonos
 - Tentar escolher planos que dividem igualmente o espaço (balanceamento da árvore)
 - Processo auxiliado pelo usuário
 - Escolha aleatória dos planos dá resultados razoáveis

Adam James. *BSP for Accelerated Hidden Surface Removal and Rendering of Static Environments*. PhD Thesis, University of East Anglia, 1999

Nodo da BSP

- ◆ O que armazenar num nodo?
 - Ponteiros para os filhos (sempre dois)
 - Ponteiro para nodo pai - útil em operações de caminhamentos
 - Se um nó folha, a extensão da região
 - Se um nó interno, o plano que divide
 - Lista de ponteiros para os conteúdos da região
 - Vizinhos? São úteis em muitos algoritmos
 - ◆ Tipicamente apenas nos nós folha
 - Portais - "buracos" que permitem ver os vizinhos

BSP em Jogos

- ◆ BSP utilizada para rapidamente eliminar partes do ambiente que não precisam ser processadas
- ◆ Esquema de visualização híbrido, que utiliza zbuffer para eliminação de elementos ocultos em conjunto com a BSP
- ◆ BSP elimina objetos do view-frustum

Demo

http://symbolcraft.com/graphics/bsp/bsptreedemo_portugese.html

BSP Resources Page

<http://cs.smith.edu/~mcharley/bsp/bsplinks.html>